(12) UK Patent Application (19) GB (11) 2 260 629 (13) A

(43) Date of A publication 21.04.1993

(21) Application No 9216731.1

(22) Date of filing 06.08.1992

(30) Priority data
(31) 777155 (32) 16.10.1991 (33) US

(71) Applicant
Intel Corporation

(Incorporated in the USA - Delaware)

2200 Mission College Boulevard, Santa Clara, California 95052, United States of America

(72) Inventors
Gary N Hammond
Pradeep Dubey

(74) Agent and/or Address for Service
Potts, Kerr & Co
15 Hamilton Square, Birkenhead, Merseyside, L41 6BR, United Kingdom

(51) INT CL^5
G06F 12/08 12/10

(52) UK CL (Edition L)
G4A AMC ANV

(56) Documents cited
GB 2176918 A GB 1428503 A EP 0272670 A2

(58) Field of search
UK CL (Edition K) G4A AMC ANV
INT CL^5 G06F 12/08 12/10
On-line database: WPI

(54) **A segment descriptor cache for a microprocessor**

(57) An improved memory segmentation system in a microprocessor, for generating a segment descriptor based on a segment selector, comprises an associative descriptor cache 570 to retain previously fetched, unscrambled, and tested descriptors for subsequent access by the same selectors. If on a memory access the descriptor is not found in the descriptor cache, the selector 501 is used to fetch a scrambled, raw descriptor from memory 530. The scrambled descriptor is then unscrambled at 550 before loading into a segment register 560. The same unscrambled descriptor is also used to update the descriptor cache 570 such that it may be found in the future. The descriptor unscrambling is necessitated by provision of backward compatibility with prior processor architectures.

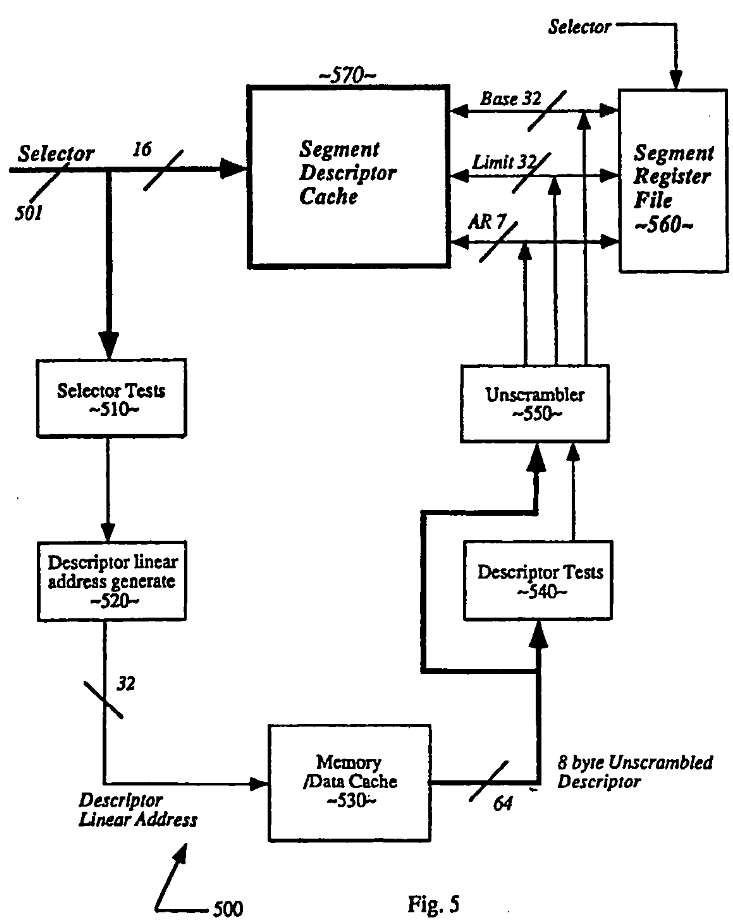


Fig. 5

At least one drawing originally filed was informal and the print reproduced here is taken from a later filed formal copy.

GB 2 260 629 A

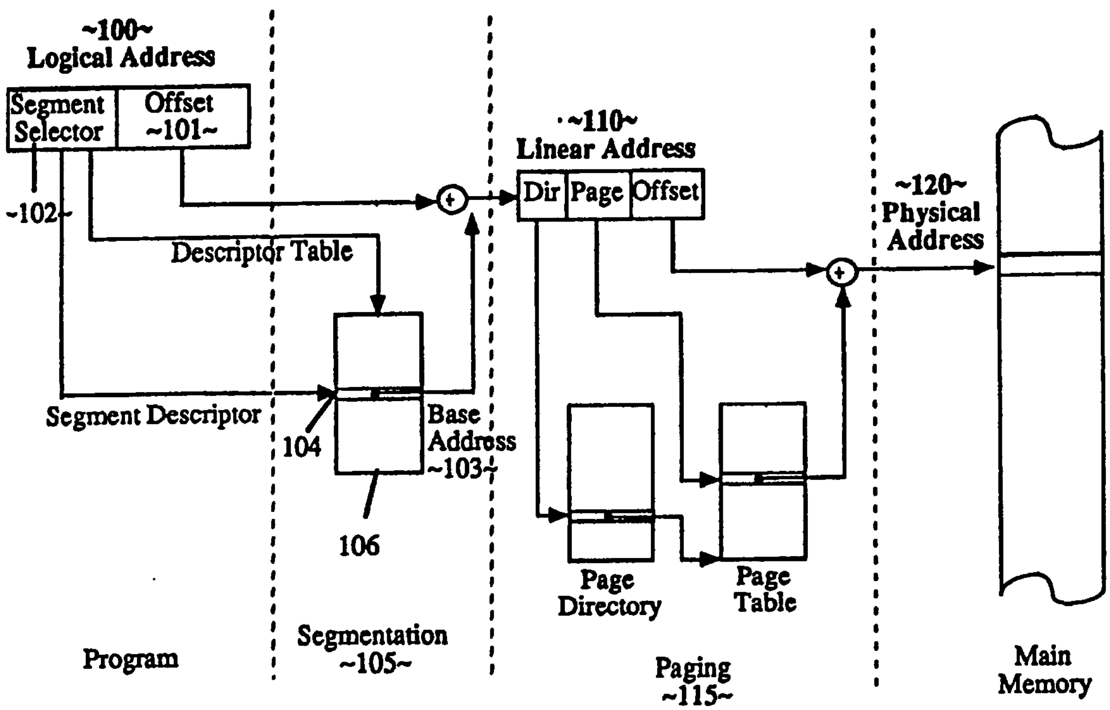

~100~
Logical Address
Segment Selector
Offset ~101~
~102~
Descriptor Table
Segment Descriptor
104
106
Base Address ~103~
~110~
Linear Address
Dir
Page
Offset
~120~
Physical Address
Page Directory
Page Table
Program
Segmentation ~105~
Paging ~115~
Main Memory

Fig. 1(a)

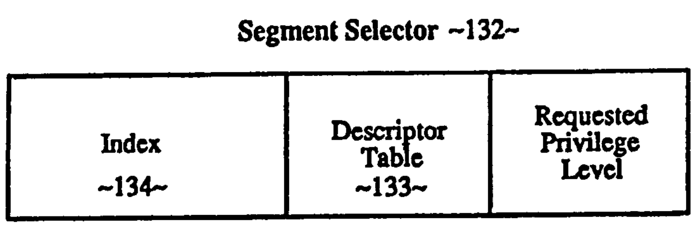

Segment Selector ~132~
Index ~134~
Descriptor Table ~133~
Requested Privilege Level

Fig. 1(b)

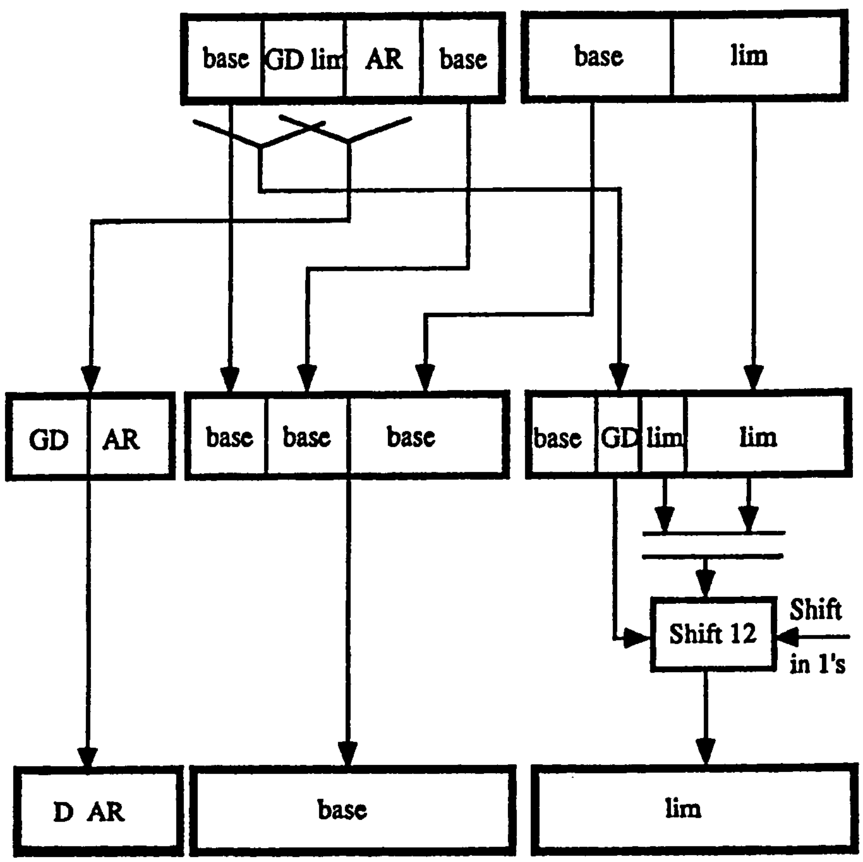
<table>
<tr><td>Base
[31:24]</td><td>G</td><td>D</td><td>0</td><td>A
V
L</td><td>Limit
[19:16]</td><td>P</td><td>D
P
L</td><td>S</td><td>Type</td><td>Base
[23:16]</td></tr>
<tr><td colspan="6">Base Address [15:0]</td><td colspan="5">Segment Limit [15:0]</td></tr>
</table>

AVL Available for Use
by System Software
Base Segment Base Address
DPL Descriptor Privilege Level
S Descriptor Type
(0 = System; 1 = Application)
G Granularity of Limit = 1 Byte/4 KBytes
Limit Segment Limit
P Segment Present
Type Segment Type
D Default Operation Size
(0 = 16-Bit Segment; 1 = 32-Bit Segment)

Fig. 2

(Prior Art)

Scrambled Segment Descriptor in Memory ~300~



Unscrambled Memory Segment Descriptor ~350~

Fig. 3

(Prior Art)

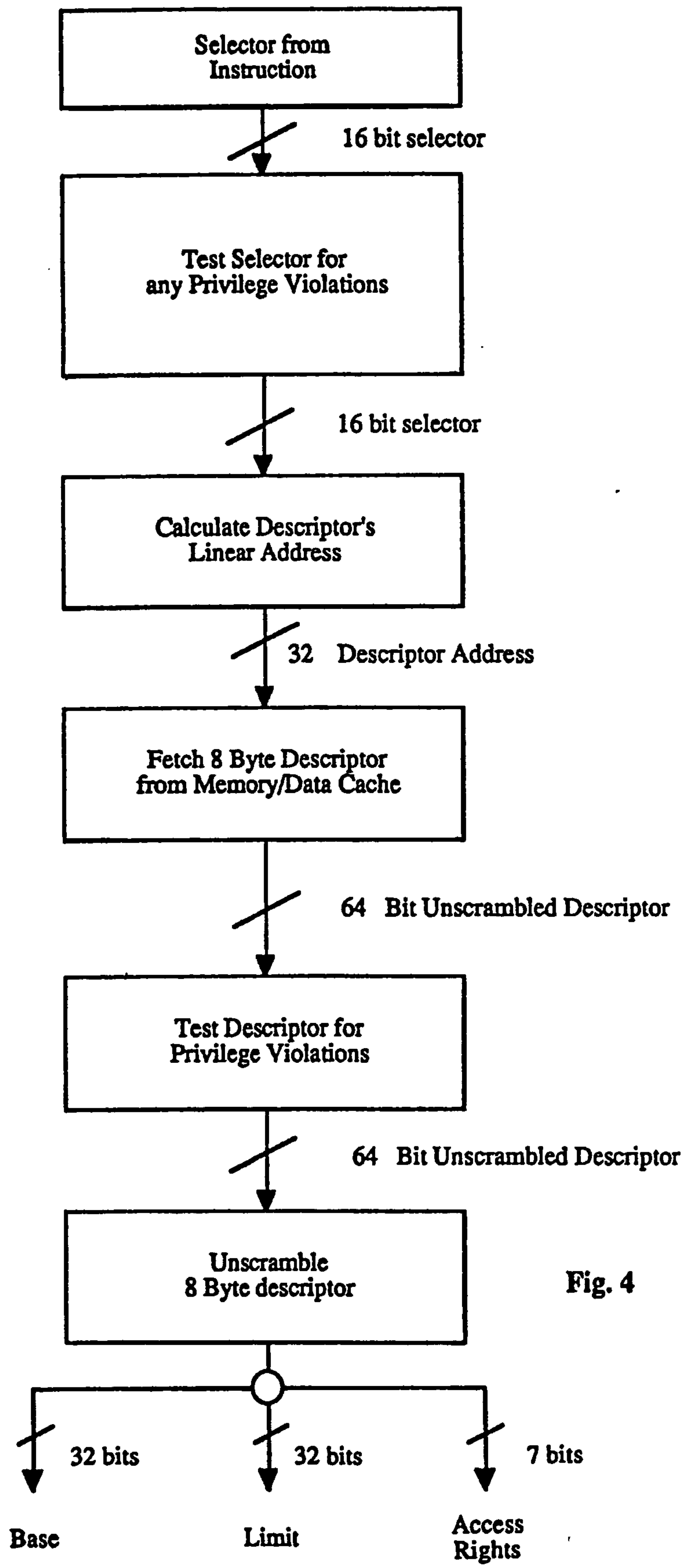

Selector from Instruction
16 bit selector
Test Selector for any Privilege Violations
16 bit selector
Calculate Descriptor's Linear Address
32 Descriptor Address
Fetch 8 Byte Descriptor from Memory/Data Cache
64 Bit Unscrambled Descriptor
Test Descriptor for Privilege Violations
64 Bit Unscrambled Descriptor
Unscramble 8 Byte descriptor
32 bits
32 bits
7 bits
Base
Limit
Access Rights

Fig. 4

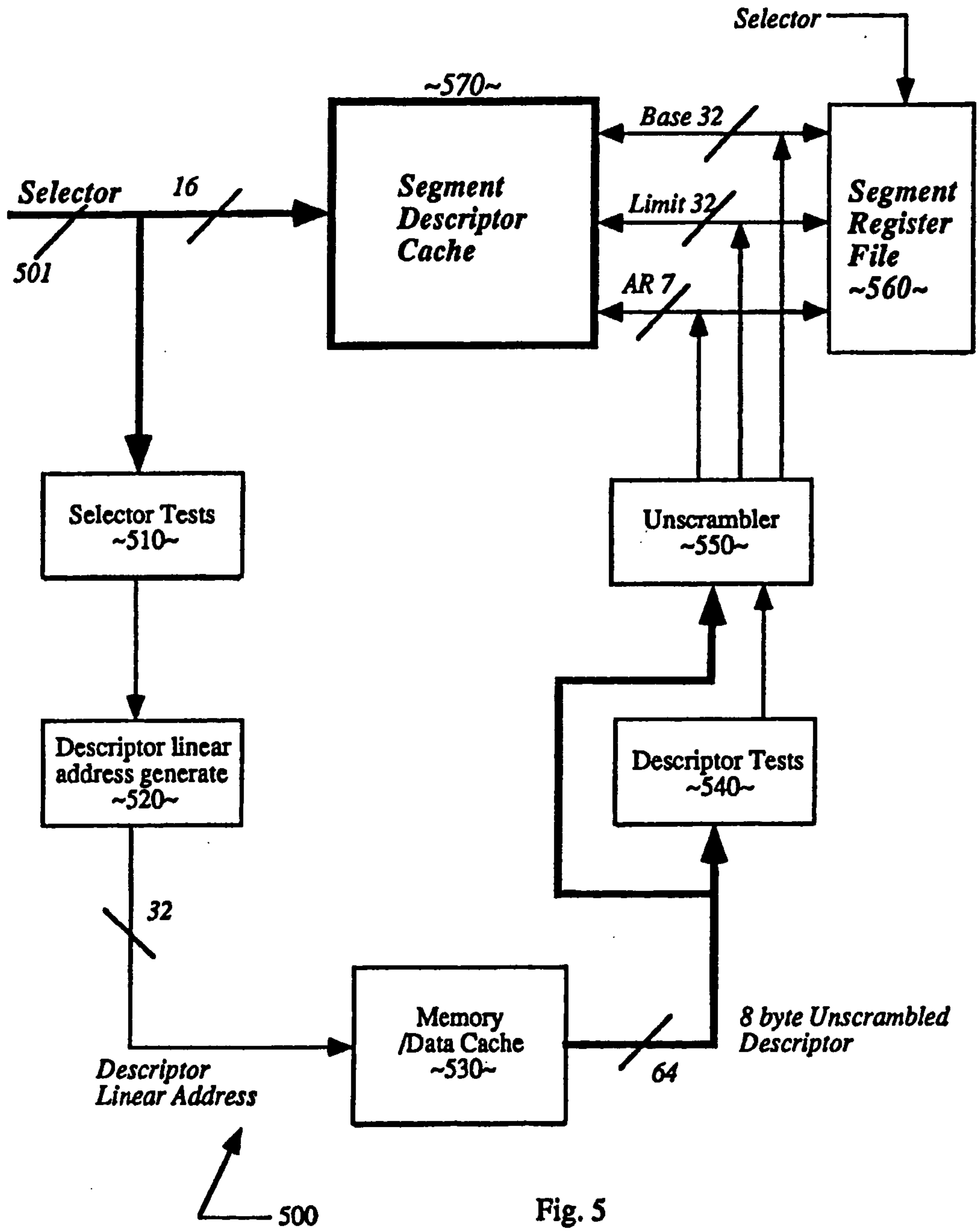

Selector
~570~
Segment
Descriptor
Cache
Base 32
Limit 32
AR 7
Segment
Register
File
~560~
Selector
16
501
Selector Tests
~510~
Unscrambler
~550~
Descriptor linear
address generate
~520~
Descriptor Tests
~540~
32
Memory
/Data Cache
~530~
64
8 byte Unscrambled
Descriptor
Descriptor
Linear Address
500

Fig. 5

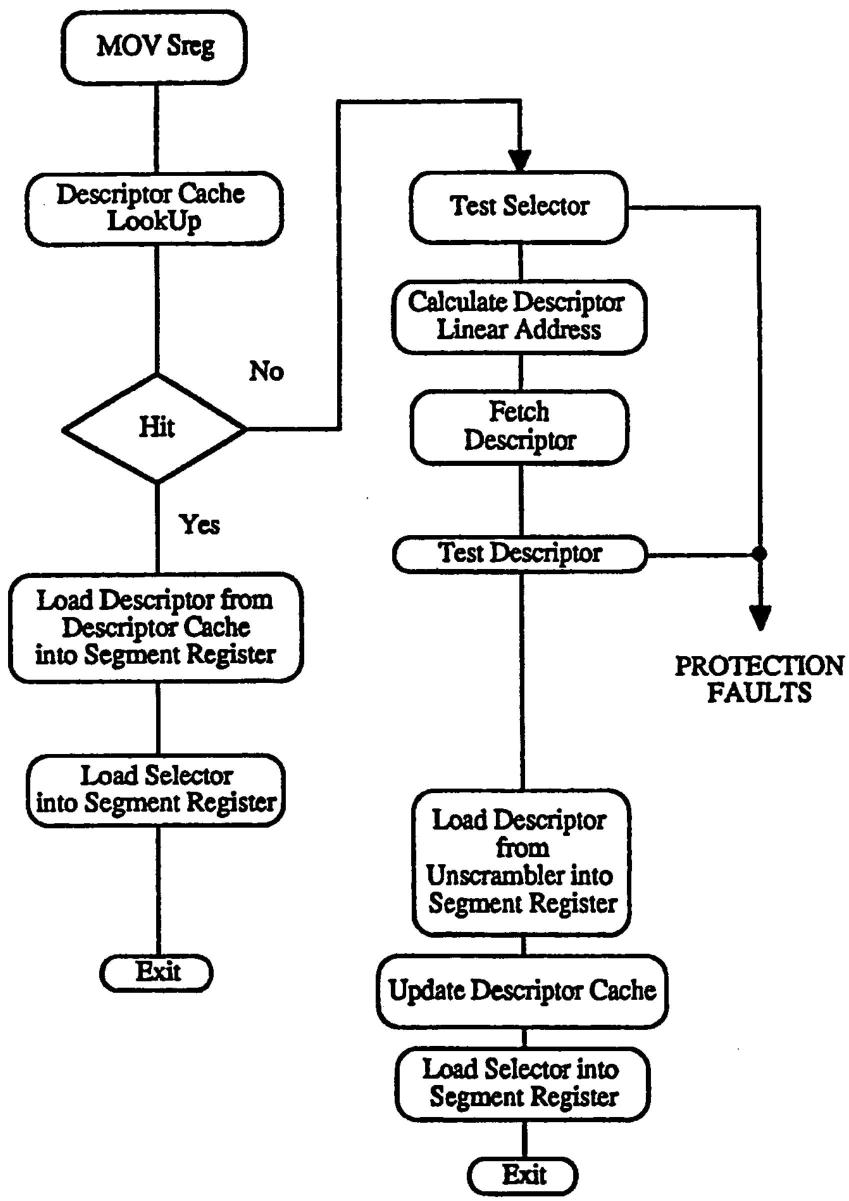

MOV Sreg
Descriptor Cache LookUp
Hit
No
Yes
Load Descriptor from Descriptor Cache into Segment Register
Load Selector into Segment Register
Exit
Test Selector
Calculate Descriptor Linear Address
Fetch Descriptor
Test Descriptor
PROTECTION FAULTS
Load Descriptor from Unscrambler into Segment Register
Update Descriptor Cache
Load Selector into Segment Register
Exit

Fig. 6

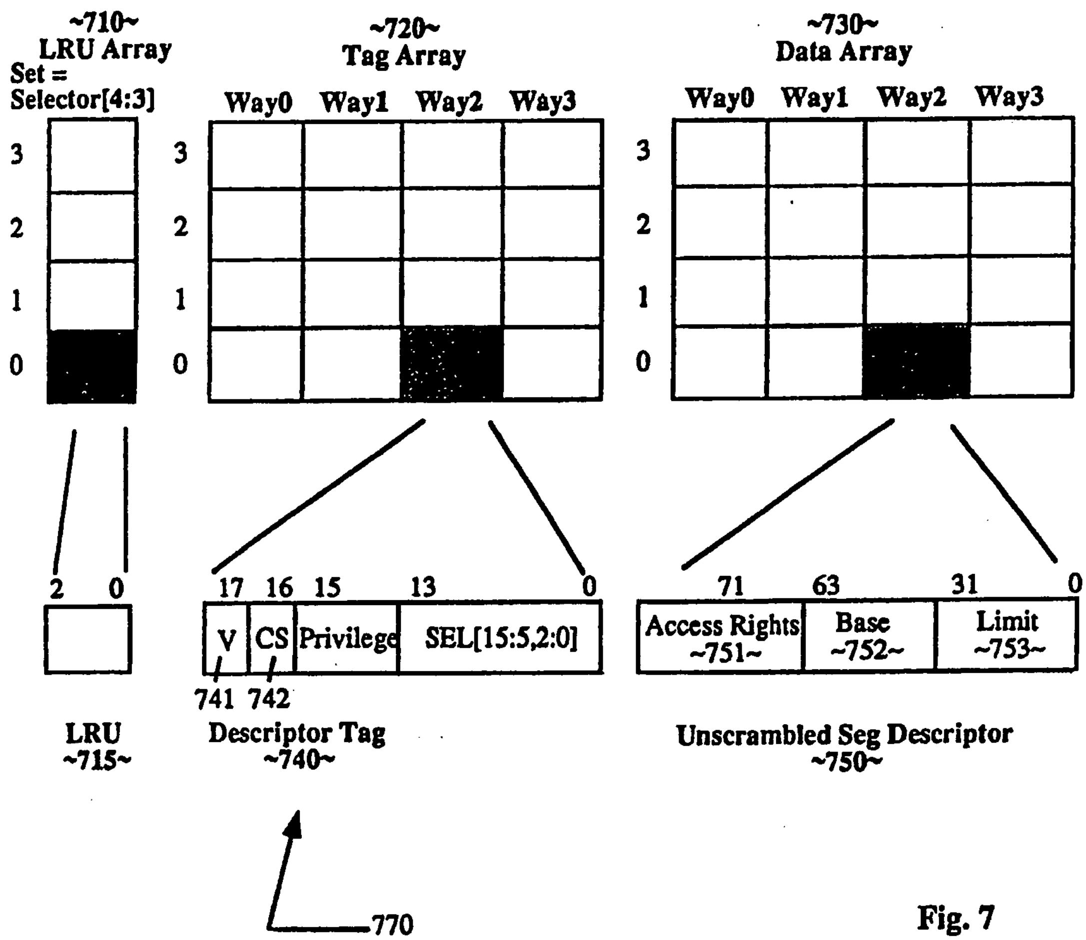

~710~
LRU Array
Set =
Selector[4:3]
3
2
1
0
~720~
Tag Array
Way0
Way1
Way2
Way3
3
2
1
0
~730~
Data Array
Way0
Way1
Way2
Way3
3
2
1
0
2
0
17
16
15
13
0
V
CS
Privilege
SEL[15:5,2:0]
741
742
71
63
31
0
Access Rights
~751~
Base
~752~
Limit
~753~
LRU
~715~
Descriptor Tag
~740~
Unscrambled Seg Descriptor
~750~
770

Fig. 7

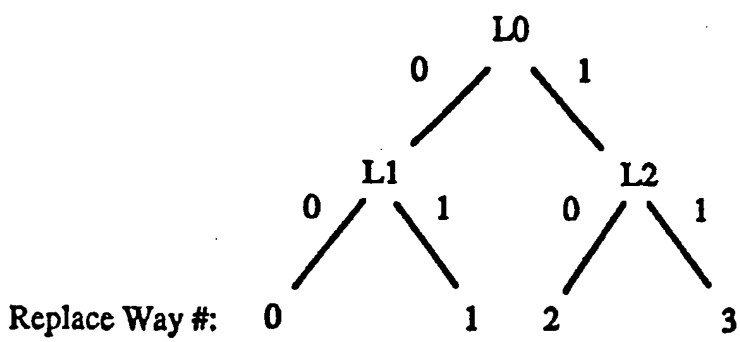


Fig. 8

2250672

A SEGMENT DESCRIPTOR CACHE FOR A MICROPROCESSOR

BACKGROUND OF THE INVENTION

1. FIELD OF THE INVENTION:

The present invention relates to memory management systems for computers and more specifically, to memory segmentation systems for microprocessors with increased data access speed and efficiency.

2. ART BACKGROUND:

Memory management is a hardware mechanism which lets operating systems create simplified environments for running programs such that when several programs are running at the same time, they may each be given an independent address space to avoid interference with each other. Memory management typically consists of segmentation and paging. Segmentation is used to give each program several independent, protected address spaces ("segments"). Paging is used to support an environment where large address spaces are simulated using a small amount of random access memory ("RAM") and some disk storage. System designers may choose to use either or both of these mechanisms. When several programs are running at the same time, either mechanism can be used to protect programs against interference from other programs.

Segmentation allows memory to be completely unstructured and simple, like the memory model of a simple 8-bit processor, or highly structured with

address translation and protection. Each segment is an independent, protected address space. Access to segments is controlled by data which describes its size, the privilege level required to access it, the kinds of memory references which can be made to it (instruction fetch, stack push or pop, read operation, write operation, etc.), and whether it is present in memory.

Reference is now made to **Fig. 1(a)**, where a pictorial representation of memory address translation mechanism is shown. Segmentation mechanism 105 translates segmented (logical) address 100 into an address for a continuous, unsegmented address space, called linear address 110. If paging is enabled, paging mechanism 115 translates linear address 110 into physical address 120. If paging 115 is not enabled, linear address 110 is used as physical address 120. Physical address 120 ultimately appears on the address bus coming out of the processor.

An example of a memory management system can be found implemented in the i486™ microprocessors manufactured by Intel Corporation of Santa Clara, California, the Assignee of the present application. In the i486™ microprocessors, a logical address consists of the 16-bit segment selector for its segment and a 32-bit offset into the segment. With reference to **Fig. 1(a)**, logical address 100 is translated into linear address 110 by adding offset 101 to base address 103 of the segment. Base address 103 is derived from segment descriptor 104, which is a data structure in memory which provides the size and location of a segment, as well as access control information. For example, the segment descriptor in a i486™ microprocessor comes from one of two tables, the global descriptor table (GDT) or the local descriptor table (LDT). There is one GDT for all programs in the system, and one LDT for each separate program or task being run. If the operating system allows, different programs can share the same LDT. The system also may be set up with no LDTs; all programs will then use the GDT. For more information with regard to the i486™ microprocessors, please refer to <u>i486™ Microprocessor: Programmer's Reference Manual</u>, available from Intel Corporation, Santa Clara, California.

The translated address is linear address 110. If paging mechanism is not used, the linear address 110 is physical address 120. If paging is used, a second level of address translation is needed to produce physical address 120.

Reference is again made to **Fig. 1(a)**. Segment selector 102 is shown pointing to segment descriptor 104 which defines a segment. A program in the i486™ microprocessors may call for more segments than those segment selectors currently occupying segment registers. When this is true, the program uses forms of MOVE instructions to change the contents of the segment registers when it needs to access a new segment. As shown in **Fig. 1(b)**, segment selector 132 identifies a segment descriptor by specifying descriptor table 133 and descriptor index 134 within that table.

Reference is now made to **Fig. 2**, where a descriptor format in the i486™ microprocessor is illustrated. However, because the descriptor format needs to provide backward compatibility for prior processor architectures, the descriptor format becomes scrambled when it is stored in memory. To simplify internal processor operations, a raw scrambled descriptor must be transformed into a unscrambled descriptor. The transformation of a scrambled segment descriptor 300 into an unscrambled segment descriptor 310 for the i486™ processors is illustrated in **Fig. 3**.

The present invention provides an improved memory management system for memory operations in microprocessors. As will be described, a segment descriptor cache is used to retain previously fetched, unscrambled, and tested descriptors such that on subsequent segment register loads, the segment descriptor can be sourced from the cache and loaded directly into the segment descriptor register file in one clock, thus bypassing all of the work and overhead usually associated with segment register loads. As will be apparent to those skilled in the art, the memory system of the present invention allows segment descriptors to be directly loaded into the segment register when the segment descriptors are cached in the segment descriptor cache after

associative searching by the segment selector. If the descriptor is not present in the descriptor cache, the selector is used to fetch the descriptor from a descriptor table in memory, unscrambles and tests the descriptor, and updates the descriptor cache with the newly fetched, unscrambled and tested descriptor such that the descriptor may be made available subsequently.

SUMMARY OF THE INVENTION

Therefore, it is an object of the present invention to provide an improved memory system to achieve greater data access speed and efficiency.

It is another object of the present invention to provide an improved memory system with a segment descriptor cache such that the previously fetched and unscrambled segment descriptors can be cached for subsequent usage.

The present invention discloses an improved memory management system in microprocessors for generating a segment descriptor in response to a segment selector. In one embodiment, the memory system of the present invention comprises a descriptor cache to retain previously fetched, unscrambled, and tested descriptors for subsequent access by the same selectors. If the descriptor is not found in the descriptor cache, the selector is used to fetch a scrambled, raw descriptor from memory. The scrambled descriptor is then unscrambled before loading into a segment register. The same unscrambled descriptor is also used to update the descriptor cache such that it may be found subsequently.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1(a) is a pictorial representation of a memory address translation mechanism.

FIGURE 1(b) illustrates the format of a segment selector.

FIGURE 2 illustrates the format of a segment descriptor.

FIGURE 3 illustrates the process of unscrambling a scrambled segment descriptor.

FIGURE 4 is a block diagram representation of the process of loading a protected mode segment register.

FIGURE 5 is a block diagram representation of a memory segmentation system incorporating the teaching of the present invention.

FIGURE 6 is a flow chart illustrating the operation of a segment descriptor load.

FIGURE 7 illustrates a currently implemented descriptor cache.

FIGURE 8 illustrates a currently implemented psuedo LRU replacement algorithm.

DETAILED DESCRIPTION OF THE INVENTION

An improved memory management system for memory operations in computers is disclosed having particular applications for use by microprocessor memory systems employing segmentation technique to enhance memory access efficiency. In the following description for purposes of explanation, specific memories, organizations, architectures, data rates, etc. are set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well known circuits are shown in block diagram form in order not to obscure the present invention unnecessarily.

Reference is now made to **Fig. 4**, where the process of loading a segment register is described in block diagram format. It should be appreciated by those skilled in the art that the segment register load is basic to any segmentation memory management scheme. As shown, when the processor encounters a selector from an instruction such as MOVE SEG REG, the processor first tests the selector for any privilege violations such that a less privileged program cannot use a more privileged program to access protected data. If no privilege violation is found, then an 8-byte descriptor is fetched by the processor from memory/data cache based on the selector's identifying a descriptor table and a descriptor within that table. This descriptor is tested for privilege violations. If no privilege violation is found, then the descriptor is unscrambled and information regarding the segment such as the size and location of the segment, as well as control and status information, can be loaded into the segment register for the program to execute.

As will be appreciated by those skilled in the art, the present invention provides an improved memory system by allowing previously fetched and tested descriptors to be retained in an associative memory. On subsequent segment descriptor loads of the same descriptor, the segment descriptor cache

is searched. If the segment descriptor cache contains the desired entry, the segment register file can be directly loaded from the segment descriptor cache, thus bypassing all the steps required in a process without the segment descriptor cache. As such, the performance of the memory segmentation system is greatly enhanced.

Reference is now made to Fig. 5, where the memory system for a segment register load incorporating the teaching of the present invention is shown. It should be understood by those skilled in the art that the dimensions of the buses are for illustrative purposes only and that the present invention can be practiced without the specific detail of the illustration. As shown in Fig. 5, system 500 comprises selector privilege tester 510 for testing any privilege violations in selector 501 such that a less privileged program cannot gain access to protected data. As used in the current implementation, when the privilege field of the segment selector contains a privilege level having a greater value (i.e. less privileged) than the program, the selector overrides the program's privilege level. If selector 501 passes privilege tester 510, it is transferred to descriptor address generator 520. Descriptor address generator 520 then generates the proper address of the descriptor in the descriptor table for the selector. Currently, two descriptor tables are used - global and local descriptor tables. The address generation process involves changing the current segment to specify a table based on the selection by selector 501, and setting the Effective Address to selector's index value. With the Effective Address, a descriptor can be fetched from memory 530 and latched into unscrambler 550 and descriptor tester 540. Descriptor tester 540 checks the descriptor to control its access to the segment. If an access violation occurs, a fault is generated to the processor. If no access violation occurs, unscrambler 550 transforms the descriptor into an internal unscrambled form as shown in Fig. 3. A successful descriptor is loaded into segment register file 560 and updated into descriptor cache 570 for future use.

Reference is now made to **Fig. 6**, where a flow chart identifying the sequence of operation of a segment descriptor load is shown. When selector 501 is encountered as in a MOV Sreg instruction, descriptor cache 570 is looked up. If a corresponding descriptor is found ("a hit"), then the descriptor is loaded from descriptor cache 570 into segment register file 560, along with selector 501. If no corresponding descriptor is found ("a miss"), then selector 501 is tested for privilege violations such that a fault may be generated for a selector with insufficient privilege level. If selector 501 passes the privilege tester 510, then descriptor linear address can be calculated and descriptor is fetched from memory 530. Descriptor is also tested for its privilege level and a fault is generated for any privilege violations. Descriptor is unscrambled and the unscrambled descriptor is loaded into segment register file 560. The unscrambled descriptor is also used to update descriptor cache 570.

Reference is now made to **Fig. 7**, where a diagram representing the currently implemented segment descriptor cache is shown. Although a 4 -by-16 set-associative cache is illustrated, it should be understood by those skilled in the art that other organizations can be easily implemented to achieve the desired functionality. Descriptor cache 770 is divided into three arrays: tag array 720, data array 730, and Least Recently Used ("LRU") array 710. The selector bits [4:3] are used as the set number and index into the descriptor cache. Each set is composed of 4 "ways", each of which is associatively searched for the desired entry.

The LRU entry 715 is composed of 3 bits used to determine which "way" in the set is least recently used. When a new entry is to be placed in the descriptor cache, the LRU entry for the set determines which entry can be replaced with a minimum of performance impact. As shown in **Fig. 8**, a psuedo LRU algorithm is used with the 3 LRU bits, called L0, L1, & L2, to determine which entry to replace.

Descriptor tag 740 from tag array 720 comprises the remaining selector bits and privilege level of the processor when the entry was placed in the descriptor cache. The tag also contains a valid bit 741 indicating if the entry is valid, and a code segment flag (CS) 742. The CS flag 742 is used to identify the type of descriptor cached, since different protection checks are applied for CS as opposed to data segments. Data array 730 contains unscrambled segment descriptors, each of which contains access rights 751, base address 752, and limit 753.

While the present invention has been described with reference to Figs. 1 through 8, it will be appreciated that the figures are for illustrative purposes only, and do not limit the spirit and scope of the invention.

CLAIMS

1. A memory segmentation system for a microprocessor, said memory segmentation system loading a segment descriptor in a second format into a segment register in response to a segment selector load instruction, said memory segmentation system comprising:

descriptor cache means for receiving said segment selector, said descriptor cache means storing a plurality of segment descriptors in said second format, each of said segment descriptors in said second format being associated with a plurality of segment selectors such that said descriptor cache means outputs a segment descriptor in said second format to said segment register if said segment descriptor in said second format is associated with said segment selector;

selector test means for receiving said segment selector when no segment descriptor in said second format in said descriptor cache means is associated with said segment selector, said selector test means determining whether said segment selector has a first predetermined privilege level;

descriptor linear address generate means coupled to said selector test means for generating a descriptor linear address from said segment selector;

descriptor-fetch means for fetching a segment descriptor in a first format from a descriptor table based on said descriptor linear address;

descriptor test means for determining whether said segment descriptor in said first format has a second predetermined privilege level; and

descriptor format means for formatting said descriptor in said first format into said second format if said segment descriptor in said first format has said second predetermined privilege level, said format means outputting said second format to said descriptor cache means to update its contents such said segment descriptor in said second format is associated with said segment

selector, said descriptor cache means also outputting said segment descriptor in said second format to said segment register,

whereby said segment descriptor in said second format is directly loaded from said descriptor cache means when said segment selector is received by said descriptor cache means.

2. A memory segmentation system according to claim 1, said descriptor cache means further comprising:

a set array comprising a plurality of sets, one of said sets being selected by said segment selector, each of said sets being replaced according to a predetermined replacement algorithm;

a tag array comprising a plurality of descriptor tags, said tag array organized into a plurality of sets, each of said sets being a plurality of ways set-associative with said sets from said set array; and

a data array comprising a plurality of segment descriptor in said second format, said data array being set-associative with each of said sets of said tag array.

3. A memory segmentation system according to claim 2, wherein said set array implements a least recently used replacement algorithm.

4. In a microprocessor memory management system, a segment of the memory being specified by a segment descriptor in a second format in a segment register, said segment descriptor in said second format being specified by a segment selector, a method of loading said segment descriptor in said second format into a segment register in response to a segment selector load instruction, said method comprising:

inputting a segment selector to descriptor cache means;

searching through said descriptor cache means to determine if a segment descriptor in a second format corresponding to said segment selector is present in said descriptor cache means, and if so, loading said segment descriptor in said second format into said segment register, said descriptor cache means comprising a plurality of segment descriptors in said second format being associated with a plurality of corresponding segment selectors;

if said segment descriptor in said second format is not present in said descriptor cache means, testing said segment selector in a selector test means to determine whether said segment selector has a first predetermined privilege level;

generating a descriptor linear address in linear address generate means;

fetching said segment descriptor in a first format from a descriptor table based on said descriptor linear address, said descriptor table storing a plurality of segment descriptors in said first format;

testing said segment descriptor in said first format in descriptor test means to determine whether said segment descriptor in said first format has a second predetermined privilege level;

formatting said segment descriptor in said first format into said second format in descriptor format means;

loading said segment descriptor in said second format into said segment register;

updating said descriptor cache means with said segment descriptor in said second format such that if the same segment selector is received by said descriptor cache means subsequently, said descriptor cache means directly causes said segment descriptor in said second format to be loaded into said segment register,

whereby a segment is specified based on said segment descriptor in said second format loaded into said segment register.

5. A method according to claim 4, wherein said descriptor cache means comprises a set array, a tag array, and a data array, said searching through said descriptor cache means further comprising:

said selector specifying a set from said set array;

selecting a tag from said tag array corresponding to said set from said set array, said tag array organized into a plurality of ways set-associative with said set array; and

selecting said segment descriptor in said second format corresponding to said tag from said tag array.

6. A method according to claim 5, further comprising replacing each of said sets from said set array according a predetermined replacement algorithm.

Pat nts Act 1977
Examin r's r p rt to th Comptroll r und r Se. 'i n 17 (The Search R port)

Application number

GB 9216731.1

Relevant Technical fields

(i) UK Cl (Edition K) G4A (AMC,ANV)

(ii) Int Cl (Edition 5) G06F (12/08, 12/10)

Search Examiner

B G WESTERN

Databases (see over)

(i) UK Patent Office

(ii) ONLINE DATABASE: WPI

Date of Search

5 OCTOBER 1992

Documents considered relevant following a search in respect of claims 1-6

Category (see over)	Identity of document and relevant passages	Relevant to claim(s)
A	GB 2176918 A (INTEL) whole document	1-6
A	GB 1428503 A (HONEYWELL) whole document	1-6
A	EP 0272670 A2 (HONEYWELL) whole document	1-6

Category	Identity of document and relevant passages	Relevant to claim(s.

Categories of documents

X: Document indicating lack of novelty or of inventive step.

Y: Document indicating lack of inventive step if combined with one or more other documents of the same category.

A: Document indicating technological background and/or state of the art.

P: Document published on or after the declared priority date but before the filing date of the present application.

E: Patent document published on or after, but with priority date earlier than, the filing date of the present application.

&: Member of th same patent family, corresponding document.

Databases: The UK Patent Office database comprises classified collections of GB, EP, WO and US patent specifications as outlined periodically in the Official Journal (Patents). The on-line databases considered for search are also listed periodically in the Official Journal (Patents).

Published 1993 at The Patent ffice, Concept House, Cardiff Road, Newport, Gwent NP9 1RH. Further copies may be obtained from Sales Branch, Unit 6, Nine Mile Point, Cwmfelinfach, Cross Keys, Newport, NP1 7HZ. Printed by Multiplex techniques ltd, St Mary Cray, Kent.